Performance Analysis of TCP variants

Rohit Aswani, Bharath Kumar D
Northeastern University
aswani.r@husky.neu.edu, dayananda.b@husky.neu.edu

***Abstract*— TCP/IP model provides a hierarchy of how data flows through the different networks. In this hyper-connected world, the allocation of network resources that includes link bandwidth, router buffers for queuing purposes pose a severe problem for competing users in the network. This leads to congestion in the network. This paper describes various TCP variants like Tahoe, Reno, New Reno and Vegas to combat against network congestion and analyses their performance under different load conditions and queuing algorithms. To analyse the behaviour of different variants, the Network Simulator -2 tool is used.**

***Keywords*— Congestion, Tahoe, Reno, New Reno, Vegas, throughput, latency, RED, Drop tail**

I. INTRODUCTION

The Internet protocol suite provides two important transport layer protocols, a connectionless protocol and a connection oriented protocol. The User Data Datagram (UDP) is a connectionless protocol while Transmission Control Protocol (TCP) is a connection oriented protocol. UDP is a fairly simple protocol and generally used for multimedia applications and does not provide reliable delivery of data between client and server. On the other hand, TCP creates a connection state on both the communicating parties and also provides in order byte reliable data delivery, flow control and congestion control.

After the formal declaration of TCP in September 1981, a plethora of improvements have been suggested and made to this protocol and many RFC's have been formulated to provide in depth explanation of TCP and its variants.

It is quite important to know and understand how each of the above mentioned TCP variant works. These variants are described below.

TCP Tahoe:
TCP Tahoe was suggested by Van Jacobson. TCP is ack-clocked and defines a congestion window variable at the sender. It uses slow start mechanism in which the sender doubles the sending rate of packets after every acknowledgment until a loss of packet occurs. Tahoe determines a loss of packet on the basis of timeout.

TCP Reno:
TCP Reno is a slight modification of TCP Tahoe, in the sense, it reacts to the loss of packet in a different manner. The algorithm is that it reduces its congestion window size to the set threshold value and increases the sending rate in a linear manner. This is called fast recovery and improves its performance due to avoidance of slow start. It also uses fast retransmit to detect three duplicate acknowledgements before the timeout occurs. It doesn't handle loss of multiple packets properly.

TCP New Reno:
TCP New Reno is an improved and more efficient variant of TCP Reno as it can detect multiple packet losses. In case of loss of multiple packets, it does not exit the fast recovery and waits until all the packets after entering the Fast recovery have been acknowledged.

TCP Vegas:
TCP Vegas uses a delay based congestion detection and control algorithm in which it proactively detects congestion in its incipient stages, thereby adjusting its throughput to prevent from packet loss. It defines congestion window on the basis of round trip times.

TCP SACK:
TCP SACK is a TCP Reno extension that provides selective acknowledgements in case of multiple packet losses. It maintains a variable pipe to estimate the number of outstanding packets in the path. This helps to solve the problems faced by TCP Reno.

In this paper, three experiments are performed on various TCP variants to observer the impact on them under the influence of load conditions, fairness between variants and lastly, their behaviour under two queuing algorithms. Since each TCP variant combats congestion in a different manner, we evaluate the performance of variants in terms of average application layer goodput, latency and packet loss. The paper is divided as follows: Section II talks about the methodology including how the experiments are performed along with the results of simulation using NS-2. Section III deals with the conclusion that includes the significance of the study and also highlights key results.

II. METHODOLOGY

In this experiment, an open source network simulator, viz. Network Simulator-2 (NS-2), is used to carry out the simulation. It facilitates with the implementation of various transport layer protocols like TCP and UDP and different queuing algorithms. NS-2 supports all the TCP variants listed in our study, making it an obvious choice of tool. The simulation results can help in determining various parameters like throughput, latency, packet loss and average bandwidth. All these parameters can be used to analyze the behavior of TCP variants under different load and queuing conditions.

Figure 1 shows the network topology used in this experiment.

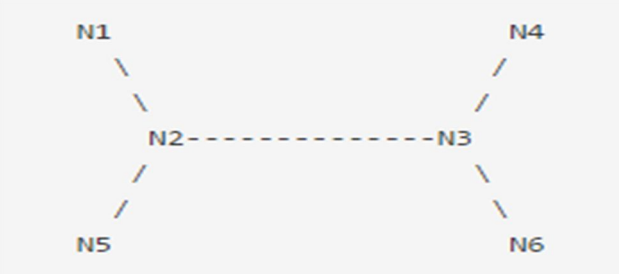

Figure 1: Network topology

The network topology consists of 6 nodes (N0, N1, N2, N3, N4, N5, N6) connected a full duplex link of bandwidth 10 Mbps and delay of 10ms. Each node uses a DropTail queuing algorithm. The CBR source is placed on node N2 and sink at node N3. The traffic sources are placed on nodes N1 and N5 while the sink nodes are placed on nodes N4 and N6. The packet size is set to 1000. CBR is considered to be an unresponsive UDP flow since it sends packets at a constant rate without caring about the packet drops. It doesn't employ congestion control. The main characteristic of CBR is that it uses a specific amount of bandwidth.

These experiments are conducted in order to evaluate the performance of different TCP variants in the presence of a constant bit rate (either variable as in first two experiments or fixed as in third experiment). The results of the simulation are analysed using the trace files generated after the simulation. These trace files are parsed using a script written in Python. The results are analysed using Microsoft Excel in order to plot the graphs of the data.

A. Experiment 1: TCP Performance Under Congestion

The objective of this experiment is to analyse the performance of TCP variants under different load conditions. The performance is evaluated in terms of goodput, packet drop rate and average latency observed as a function of bandwidth of the CBR flow. This means, the bandwidth is varied at the rate of 1 Mbps. In this experiment, the CBR source is added at N2 and a sink at N3. The TCP stream flows from N1 to N4.

Figure 2 shows the result of goodput ((total number of received packets * number of bits per byte) / (total time * 1000) versus the constant bit rate for TCP variants. As shown in figure, the TCP variant New Reno starts with a higher value of throughput due to low traffic and performs well initially whereas TCP Tahoe and Reno shows almost the same behaviour due to slow start mechanism, probing for bandwidth. TCP Vegas starts with a low value of throughput since it calculates the base RTT and compares it with the most recent RTT value. In other words, Vegas basically waits to know the network resources i.e. available bandwidth. This mechanism is called Modified Slow Start which is used by Vegas. With increase in CBR rate, all the TCP variants perform the same until the CBR rate is between 6 and 7 Mbps. At this point, the congestion is high in the network, thereby causing the reduction in the values of throughput. With further increase in CBR rate, it can be seen that TCP Tahoe throughput drops because of coarse grain timeout in case of packet loss. The fall in throughput for Reno and New Reno is almost the same and later New Reno shows better performance due to not exiting the fast recovery mode until all the packets in the outstanding path are acknowledged. At the same time, TCP Vegas has outperformed in this case due to its different congestion control algorithm which uses calculation of RTT to detect congestion instead of packet loss.

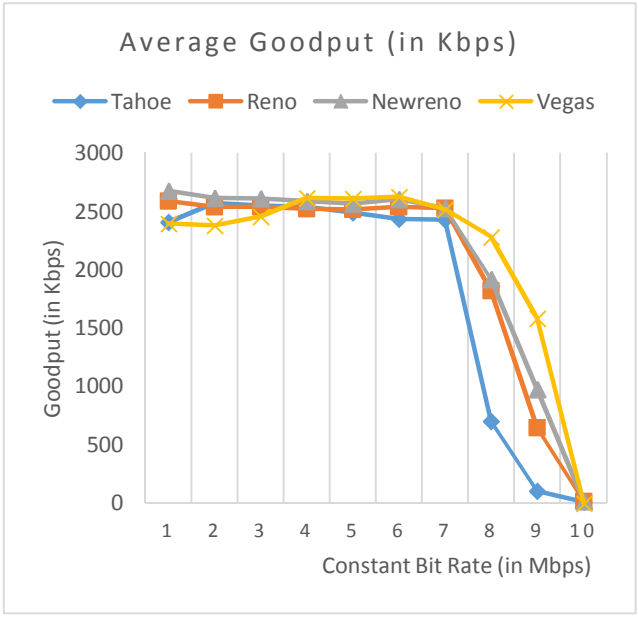

Figure 2: Goodput versus CBR

Figure 3 shows the result of packet drop rate versus CBR. As shown in the figure, the packet drop rate is identical for all the variants till 8 Mbps. After 8 Mbps, due to congestion, the packet drop rate becomes high for Tahoe. Reno and New Reno have low packet loss than Tahoe, eventually New Reno showing less packet drop rate than Reno. Vegas, due to its delay based congestion control algorithm, shows very less packet drop rate. These results are portrayed in Figure 3.

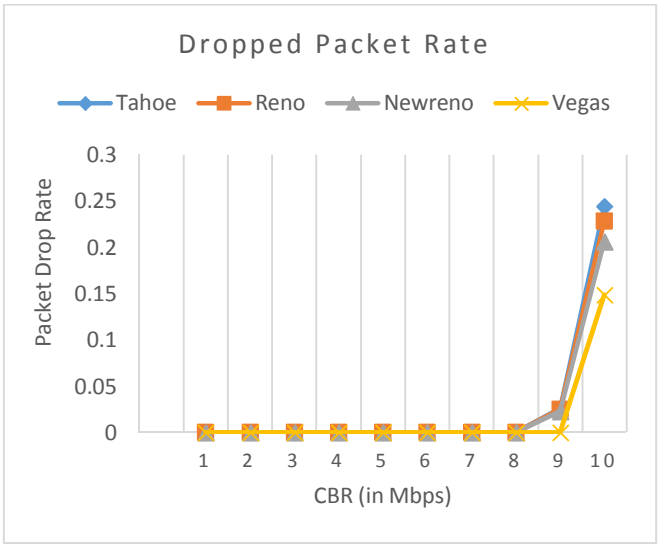

Figure 3: Packet Drop rate versus CBR

Figure 4 shows the result of latency versus CBR rate for various TCP variants. As seen in figure 4, the latency for all variants in initial stage is almost the same. When the congestion is observed, TCP Vegas shows the least latency because it proactively detects congestion in the network using the calculation of RTT and doesn't use the loss of packet as an indication to detect congestion. Other variants show high latency due to use of changing congestion window sizes in case of packet loss.

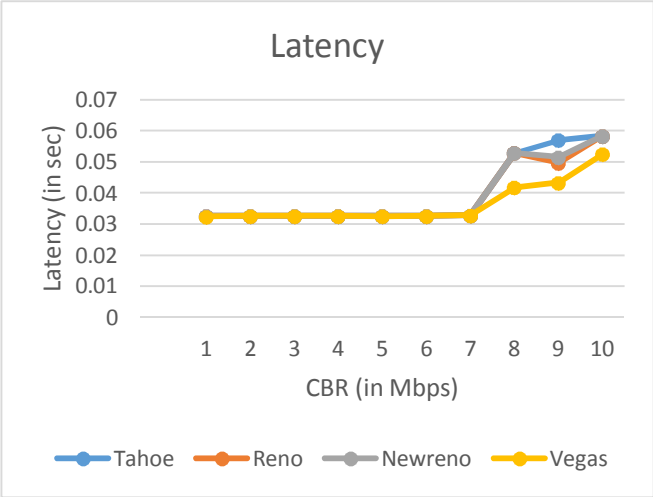

Figure 4: Latency versus CBR rate

B. *Experiment 2: Fairness Between TCP Variants*

In this experiment, the analysis if fairness between various TCP variants is done. The CBR source is at N2 and sink at N3. Two TCP flows are from N1 to N4 and N5 to N6. The results are used to calculate average throughput, packet loss rate, and latency of each TCP flow by varying the CBR flow rate. This experiment is conducted between these two pairs of variants, namely, Reno/Reno, New Reno/ Reno, Vegas/ Vegas and New Reno/ Vegas.

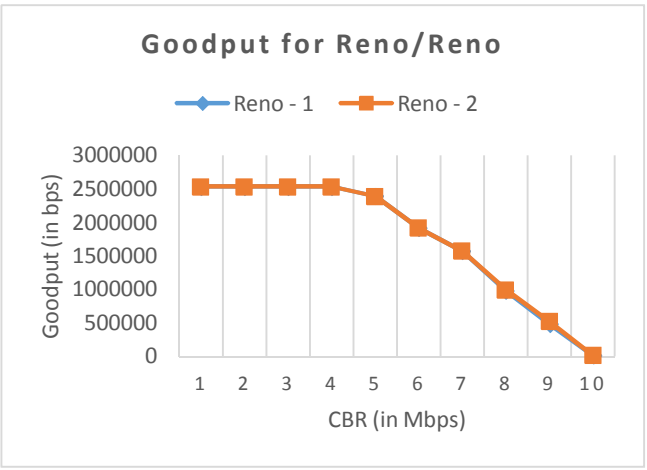

Figure 5: Graph of goodput for Reno/Reno

Figure 5 shows the graph for goodput for Reno/ Reno. It depicts that that these two variants are fair to each other in terms of bandwidth because of the same congestion control algorithm. With increase in CBR rate, these two variants show same goodput values.

Figure 6 shows the fairness comparison between New Reno and Reno. As it can be seen, New Reno and Reno have the same throughput value till 7 Mbps. After the onset of congestion, New Reno uses more bandwidth, leading to unfair share of bandwidth. The reason for this behaviour is that New Reno can handle multiple packet loss as compared to Reno. Also, it doesn't exit the fast recovery mode until all the packets are acknowledged when it was initiated.

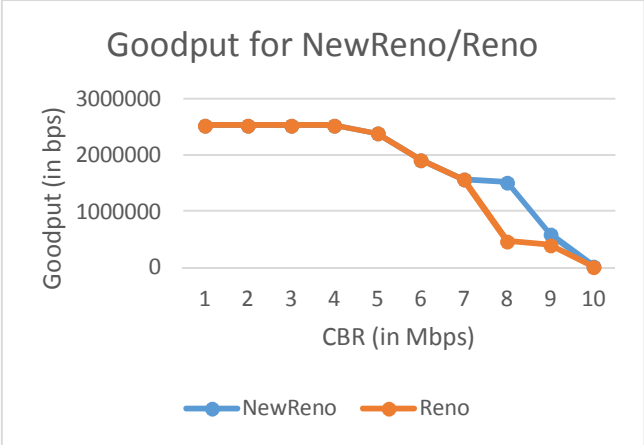

Figure 6: Graph of goodput for New Reno / Reno

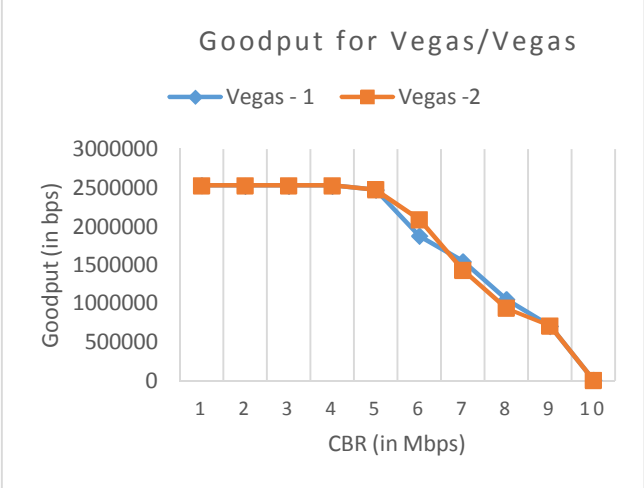

Figure 7: Graph of goodput for Vegas/Vegas

Figure 7 shows the goodput comparison between Vegas and Vegas for two TCP flows. As it can be seen, the goodput for both the flows is almost the same before the congestion occurs in the network. After the congestion at 5 Mbps, Vegas-1 provides fair share of bandwidth due to the other flow. It reduces its sending rate when the network is congested and makes use of available bandwidth when the congestion is low. This can be easily seen from the graph i.e. when goodput for Vegas 1 is low, the goodput for Vegas 2 is

better and vice versa. However, the overall goodput almost remains the same.

Figure 8 shows the comparison of goodput for New Reno and Vegas. As seen from the graph, the goodput for both the variants almost the same until congestion occurs where New Reno occupies r bandwidth, thereby providing high goodput. Vegas is fair to New Reno by means of conservative approach as it adjusts its sending rate on the basis of RTT and available bandwidth in case of high congestion.

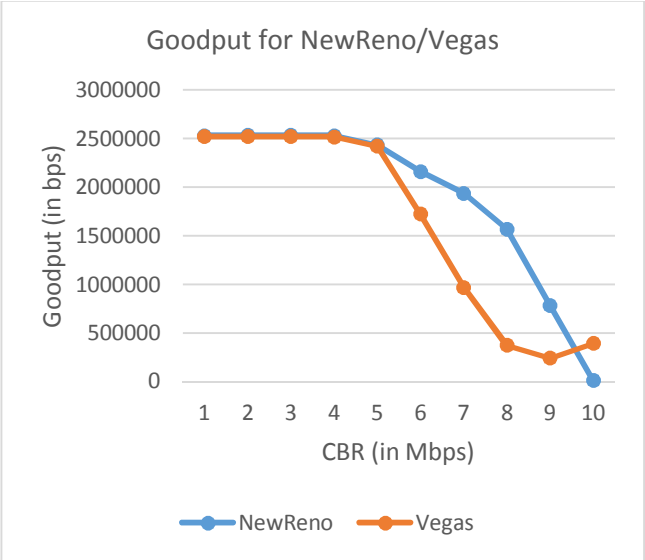

Figure 8: Graph for goodput of New Reno and Vegas

C. *Experiment 3: Influence of Queuing*

In this experiment, the behaviour of TCP variants (Reno and Sack) is evaluated by the use of queuing algorithms like DropTail and Random Early Detection and checking its impact on application layer throughput for two flows.

This experiment uses one TCP from node N1 to N4 and one CBR flow N5 to N6. First, the TCP flow is started, the time is allowed to elapse until it achieves the steady state and then the CBR flow is started. The performance of DropTail and RED is analysed for both the flows. The graph is plotted between the performance of TCP and CBR over time.

Figure 9 shows the graph for goodput versus time for Sack under the influence of RED and DropTail. As it can be seen, the goodput of Sack under RED and DropTail is almost the same till the time CBR flow is injected into the network. On the onset of CBR, the TCP flow lowers and hence goodput drops dramatically for both the queuing algorithms. However, the goodput is less for Sack (RED) than Sack (DropTail) due to the property of RED being unbiased towards the bursty traffic when CBR flow enters the network. This also means that RED is more fair than its corresponding competent DropTail [5].

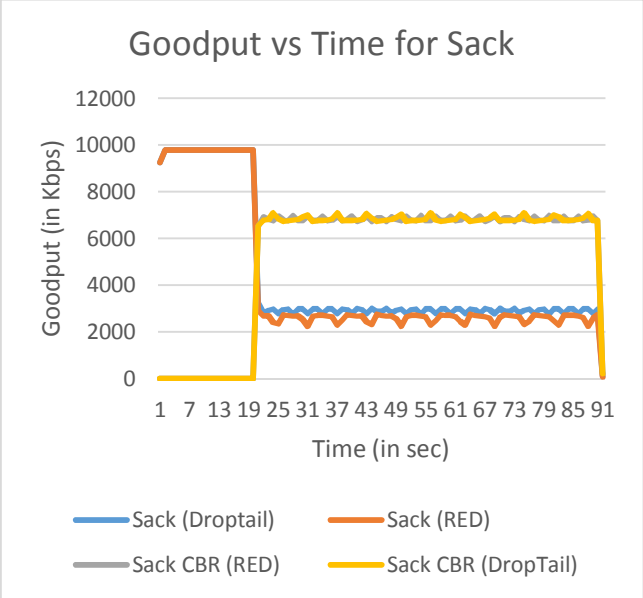

Figure 9: Graph of goodput vs time for Sack (RED) and Sack (DropTail)

It can also be seen that RED with Sack is a better idea as it does not behave in a biased manner when it comes to bursty traffic. This leads to improvement in the performance. The other reason is due to the property of Sack to avoid unnecessary re missions which leads to better throughput. Random Early Detection provides fairness in terms of bandwidth than DropTail. The reason for this is that RED uses statistical probability approach when dropping the packets, i.e. if the host transmits at a high rate, it's highly likely that the packets will get dropped in proportion to the amount of data in the queue [5].

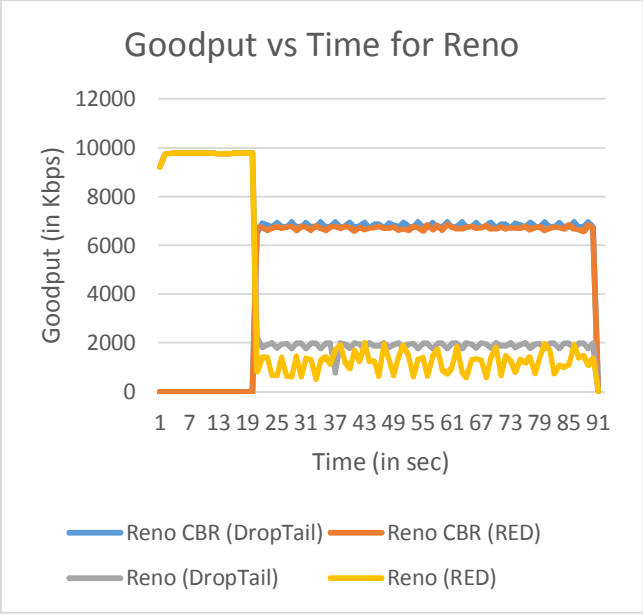

Figure 10: Graph of goodput vs time for Reno (RED) and Reno (DropTail)

Figure 10 shows that the output of goodput for Reno under the influence of DropTail and RED. In order to determine the fairness between two queuing algorithms for the same variant, we see that the goodput for Reno with DropTail is high than with RED. This means that RED is quite fair as compared to DropTail.

In case of end to end latency and drop rate for both the flows, the values are tabulated as below:

	DropTail /Sack	DropTail / Reno	RED / Sack	RED / Reno
Latency (sec)	0.03770	0.03686	0.03371	0.03381
Drop Rate (%)	0.84622	0.73844	0.43378	0.60161

Table 1: Latency and drop rate values

Table 1 shows the values for latency and drop rate for Reno and Sack under the influence of DropTail and Random Early Detection. As seen clearly from the table above, the latency is quite high in case of DropTail used for Reno and Sack while the latency in case of RED is quite low. The reason behind this is due to probabilistic nature of dropping packets used by RED queuing algorithm rather than handling bursty traffic used by DropTail. When RED is used with Sack, we see that the latency is lowest, thereby giving another reason to choose RED with Sack. With respect to drop rate, RED shows less number of packet loss as compared to DropTail, eventually decreasing the end to end latency.

III. Conclusion

Various TCP variants show different performance under the influence of congestion, fairness and queuing algorithms. In this paper, the results for three different experiments on TCP variants is presented.

In the first experiment, the behaviour of TCP variants is analysed under various load conditions is evaluated. Our results shows that TCP Vegas outperforms all the other variants as it provides higher goodput, low packet loss and less latency. Thus, TCP Vegas should be a choice of variant in highly congested networks.

The second experiment analyses the fairness between various variants. Our results shows that the pair of variants, namely, Reno/Reno and Vegas/Vegas depict fairness between the pair itself. Thus, we conclude that fairness of bandwidth exists only when the two flows use the same TCP variant in the network. Use of different TCP variants as a combination leads to unfair utilization of the bandwidth.

In the third experiment, the analysis of different queuing mechanisms is performed on two TCP variants. The results indicate that Random Early Detection is better as it provides fairness of the bandwidth and less latency due to its unbiasness towards to bursty packet traffic.

References

[1] A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas. http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf

[2] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993

[3] Kevin Fall and Sally Floyd "Simulation-based Comparisons of Tahoe, Reno and SACK TCP"

[4] Lawrence S. Brakno, Sean W.O'Malley, Larry L. Peterson "TCP Vegas: A New Techniques for Congestion Detection And Avoidance SIGCOMM/ACM, 1994

[5] Random Early Detection: http://en.wikipedia.org/wiki/Random_early_detection